R E P U B L I Q U E F R A N Ç A I S E PCT/FR 01/00166

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

FR01/166

4

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

REÇU 01 MARS 2001
WIPO PCT

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 31 JAN. 2001

Pour le Directeur général de l'Institut national de la propriété industrielle
Le Chef du Département des brevets

M Planche

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS CONFORMÉMENT À LA RÈGLE 17.1 a) OU b)

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30
http://www.inpi.fr

BREVET D'INVENTION

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

26bis, rue de Saint-Pétersbourg
75800 Paris Cédex 08
Téléphone: 01 53.04 0001033 Télécopie: 01.42.94.86.54

Code de la propriété intellectuelle-livreVI

REQUÊTE EN DÉLIVRANCE

DATE DE REMISE DES PIÈCES 26.01.00 N° D'ENREGISTREMENT NATIONAL DÉPARTEMENT DE DÉPÔT 99 DATE DE DÉPÔT 26.01.00	GEMPLUS Avenue du Pic de Bertagne Parc d'activités de Gèmenos 13881 GEMENOS France
Vos références pour ce dossier: GEM0781	

1 NATURE DE LA DEMANDE			
Demande de brevet			
2 TITRE DE L'INVENTION			
	ALGORITHME D'EXPONENTIATION MODULAIRE DANS UN COMPOSANT ELECTRONIQUE METTANT EN OEUVRE UN ALGORITHME DE CHIFFREMENT A CLE PUBLIQUE		
3 DECLARATION DE PRIORITE OU REQUETE DU BENEFICE DE LA DATE DE DEPOT D'UNE DEMANDE ANTERIEURE FRANCAISE	Pays ou organisation	Date	N°
4-1 DEMANDEUR			
Nom	GEMPLUS		
Suivi par	BRUYERE Pierre		
Rue	Avenue du Pic de Bertagne Parc d'activités de Gèmenos		
Code postal et ville	13881 GEMENOS		
Pays	France		
Nationalité	France		
Forme juridique	Société anonyme		
N° SIREN	349 711 200		
Code APE-NAF	321B		
N° de téléphone	04.42.36.69.06		
N° de télécopie	04.42.36.63.43		
Courrier électronique	nathalie.herail@gemplus.com		
6 DOCUMENTS ET FICHIERS JOINTS	Fichier électronique	Pages	Détails
Désignation d'inventeurs			
Description	ppad gem 781.doc	12	
Revendications	ppad gem 781.doc	3	5
Abrégé	ppad gem 781.doc	1	
Listage de séquences			
Rapport de recherche			
7 RAPPORT DE RECHERCHE			
Etablissement immédiat			

8 REDEVANCES JOINTES	Devise	Taux	Quantité	Montant à payer
062 Dépôt	FRF	250.00	1.00	250.00
063 Rapport de recherche (R.R.)	FRF	2 100.00	1.00	2 100.00
Total à acquitter	FRF			2 350.00

9 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE	
Signé par	GEMPLUS Bernard NONNENMACHER

La loi n°78-17 du 6 janvier 1978 relative à l'informatique aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

Référence EASY : 69444

INPI INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

Désignation de l'inventeur

0001333

Vos références pour ce dossier	GEM0781
N°D'ENREGISTREMENT NATIONAL	
TITRE DE L'INVENTION	
	ALGORITHME D'EXPONENTIATION MODULAIRE DANS UN COMPOSANT ELECTRONIQUE METTANT EN OEUVRE UN ALGORITHME DE CHIFFREMENT A CLE PUBLIQUE
LE(S) DEMANDEUR(S) OU LE(S) MANDATAIRE(S):	

DESIGNE(NT) EN TANT QU'INVENTEUR(S):	
Inventeur 1	
Nom	BENOIT
Prénoms	Olivier
Rue	22 rue Rastegue
Code postal et ville	13400 AUBAGNE
Société d'appartenance	GEMPLUS

DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE	28/01/00 R. NONNENMACHER

La présente invention concerne un algorithme d'exponentiation modulaire anti SPA, de l'anglais " Simple Power Attack " dans un composant électronique mettant en œuvre un algorithme de chiffrement à clé publique.

Les caractéristiques des algorithmes de cryptographie à clé publique sont connus : calculs effectués , paramètres utilisés. La seule inconnue est la clé privée contenue en mémoire programme. Toute la sécurité de ces algorithmes de cryptographie tient dans cette clé privée contenue dans la carte et inconnue du monde extérieur de cette carte. Cette clé privée ne peut être déduite de la seule connaissance du message appliqué en entrée et du message chiffré fourni en retour ou de la connaissance de la clé publique.
Or il est apparu que des attaques externes, basées sur les consommations de courant ou une analyse de consommation en courant lorsque le microprocesseur d'une carte est en train de dérouler l'algorithme de cryptographie pour signer un message, déchiffrer un message, permettant à des tiers mal intentionnés de trouver la clé privée contenue dans cette carte.
Ces attaques sont appelées attaques SPA, acronyme anglo-saxon pour Single Power Analysis.
Le principe de ces attaques SPA repose sur le fait que la consommation de courant du microprocesseur exécutant des instructions varie selon la donnée manipulée.
Notamment, quand une instruction exécutée par le microprocesseur nécessite une manipulation d'une donnée bit par bit, on a deux profils de courant

différents selon que ce bit vaut " 1 " ou " 0 ". Typiquement, si le microprocesseur manipule un " 0 ", on a à cet instant d'exécution une première amplitude du courant consommé et si le microprocesseur manipule un " 1 ", on a une deuxième amplitude du courant consommé, différente de la première.
Ainsi l'attaque SPA exploite la différence du profil de consommation en courant dans la carte pendant l'exécution d'une instruction suivant la valeur du bit manipulé. D'une manière simplifiée, la conduite d'une attaque SPA consiste à identifier une ou des périodes particulières du déroulement de l'algorithme comprenant l'exécution d'au moins une instruction manipulant des données bit par bit et à distinguer deux profils de consommation de courant différents, l'un correspondant à la manipulation d'un bit égal à " 0 " et l'autre correspondant à un bit égal à " 1 ". L'analyse s'effectue sur une courbe ou éventuellement sur n courbes du même déroulement de l'algorithme moyenné pour supprimer le bruit.

L'exponentiation modulaire est défini par la formule mathématique suivante :
$R = X^Y \bmod N$,

dans laquelle :

Y est un exposant qui a une taille de k bits ;
N est un modulus qui a une taille de k' bits.

X est une variable connue qui a une taille de k'' bits ;

R est le résultat de l'opération d'exponentiation modulaire et a une taille de k' bits.

On peut utiliser les algorithmes classiques A ou B connus et décrits ci-dessous.

L'algorithme classique A utilisé pour le calcul de la formule mathématique ci dessus mentionnée est le suivant :

- On initialise R à 1 : R=1 ;
- On parcours la représentation binaire de Y du bit de poids fort noté Y(k-1) vers le bit de poids faible Y(0) ;

  pour chaque bit Y(i), i variant de (k-1) à 0, on effectue l'opération supplémentaire : $R=R^2$.

  Si le bit Y(i) est égal à 1, une étape supplémentaire est exécutée qui consiste en l'opération :

  $$R=R*X.$$

Si par exemple Y est égal à 5, sa représentation binaire est : 101 ;

Si on applique l'algorithme ci-dessus :

- pour le premier bit [Y(2)=1], on effectue $R=R^2$ suivi de l'opération : R*X=X, soit le résultat R=X ;
- pour le second bit [Y(1)=0], on effectue l'opération $R=R^2$ soit le résultat $R=X^2$;
- pour le troisième bit [Y(0)=1], on effectue l'opération $R=(R^2)^2$ suivi de

l'opération $R=R*X$ soit le résultat : $R=(X^2)^2=X^5$.

Pour rappel, on reprend toujours le R précédent.

effectue l'operation : $Z=Z=X$ et

Bien sûr, toutes les opérations mathématiques décrites pour l'exemple Y est égal à 5 sont effectuées modulo N ce qui permet de travailler avec un registre r d'une taille de k' bits.

L'algorithme classique B utilisé pour le calcul de la formule mathématique ci-dessus mentionnée est le suivant :

- on initialise R à 1 et Z à X : $R=1$ et $Z=X$, Z étant une variable :
- on parcours la représentation binaire de Y du bit de poids faible $Y(0)$ vers le bit de poids fort $Y(k-1)$;

  pour chaque bit $Y(i)$, i variant de 0 à $(k-1)$, on effectue l'opération supplémentaire : $Z=Z^2$, quelque soit i supérieur à 0 ;

  si le bit $Y(i)$ est égal à 1, une étape supplémentaire est exécutée qui consiste en l'opération : $R=R*Z$.

Si par exemple Y est égal à 5, sa représentation binaire est 101.

Si on applique l'algorithme ci-dessus :

- pour le premier bit, on a $Y(0)=1$; on n'effectue pas l'opération Z^2 (car $i=0$) et on effectue l'opération : $R=R*Z=X$.

- pour le second bit, on a [Y(1)]=0, on effectue l'opération $Z^2=X^2$; R est inchangé car Y(1)=0 ;
- pour le troisième bit [Y(2)=1], on effectue l'opération : $Z=Z^2=X^4$ et comme Y(2) est égal à 1, on fait aussi l'opération R=R*Z et on obtient donc X^5.

Pour rappel, on reprend toujours le R et le Z précédent.

Bien sûr, toutes les opérations mathématiques décrites pour l'exemple Y est égal à 5 sont effectuées modulo N ce qui permet de travailler avec des registres r et z d'une taille de k' bits.

Cependant, cet algorithme B est rarement utilisé dans un composant électronique de type carte à puce car il nécessite plus de mémoire (registre supplémentaire z d'une taille de k' bits).

On constate que, sur les algorithmes classiques A et B expliqués ci-dessus, en fonction de chaque bit de Y on effectue une opération si le bit est égal à 0 et deux opérations si le bit est égal à 1. Ces algorithmes A et B sont utilisés pour le RSA. Pour rappel, le système de chiffrement RSA est le système de chiffrement à clé public le plus utilisé. Il peut être utilisé comme procédé de chiffrement ou comme procédé de signature. Le système de chiffrement RSA est utilisé dans les cartes à puce pour certaines applications de celles-ci. Les applications possibles de RSA sur une carte à puce sont

l'accès à des banques de données, des applications bancaires, des applications de paiements à distance, comme par exemple la télévision à péage, la distribution d'essence ou le paiement de péages d'autoroute. Cette liste d'applications est bien sûr non exhaustive.

Le principe du système de chiffrement RSA est le suivant. Il peut être divisé en trois parties distinctes qui sont :

1) La génération de la paire de clés RSA ;

2) Le chiffrement d'un message clair en un message chiffré, et

3) Le déchiffrement d'un message chiffré en un message clair.

Une opération RSA de chiffrement consiste en ce qu'on calcule un chiffré c qui est égal à un message M^e mod N représenté par l'opération :$C= M^e$ mod N, dans laquelle e est l'exposant publique de chiffrement et N est le modulus.

Une opération RSA de déchiffrement consiste en ce qu'on calcule un message M' qui est égal à M si on déchiffre de manière juste et est représenté par l'opération:

$M' = C^d$ mod N,

dans laquelle d est l'exposant privé de déchiffrement et N le modulus.

On constate que le RSA est directement une opération d'exponentiation modulaire.

Il s'avère que d est un élément secret puisque privé ; on constate donc que d est équivalent à Y de l'algorithme classique A ou B, algorithmes décrits au début de la description. Cependant,

ces algorithmes utilisés pour le RSA peuvent être attaqués par simple étude de la consommation de courant du composant électronique mettant en œuvre l'invention.

En effet, si on considère que la signature S d'une opération R^2 pour l'algorithme A et Z^2 pour l'algorithme B appelée " operation square ", notée S(SQU), est différente de la signature S de l'opération R*X pour l'algorithme A et Z*R pour l'algorithme B, appelée " operation multiply ", notée S(MUL), alors la consommation de courant, lors de l'exécution de l'algorithme A ou B décrits précédemment, consiste en une succession de signatures S(SQU) et S(MUL) directement dépendante de Y.

Par exemple, dans le cas de l'algorithme A, pour Y égal à 5, on aura la succession suivante de signatures :

[S(SQU), S(MUL)], [S(SQU)], [S(SQU), S(MUL)], dans laquelle succession les signatures [S(SQU)] suivie de [S(MUL)] correspond à un bit égal à 1 et la signature [S(SQU)] suivi de la signature [S(SQU)] correspond à un bit égal à 0.

Simplement en regardant la consommation de courant si on sait différencier S(SQU) de S(MUL), on peut retrouver l'intégralité de la valeur Y. Si on applique cette attaque au RSA décrit ci-dessus, on retrouve Y=d qui est l'exposant privé de déchiffrement qui doit rester secret par définition ce qui est donc très gênant.

La présente invention permet de supprimer cet inconvénient majeur.

Cependant, pour bien insister sur l'inventivité de la présente invention, il est utile de décrire un exemple d'amélioration des algorithmes A et B pourtant défaillants.

Dans l'algorithme classique A ou B, on considère que le composant qui met en œuvre l'invention dispose d'une opération optimisée appelée " Square ", notée SQU qui calcule R^2 de manière plus efficace que l'opération " Multiply ", notée MUL.

La première parade contre l'attaque consiste à utiliser uniquement l'opération MUL. Dans ce cas, il ne subsiste plus que la signature de l'opération " Multiply " ce qui ne permet plus de distinguer une quelconque information permettant de remonter à la valeur Y. Plus précisément, l'opération mathématique " Multiply " a deux opérandes V et W et est définie par la formule :

MUL(V,W) = V*W.

En théorie, on se protège mais dans la pratique on utilise l'opération MUL(V,V) ou l'opération MUL(V,W) ; il y a donc encore une différence dans la consommation de courant puisque les opérandes sont différents. Il ne s'agit pas d'une solution fiable.

La présente invention consiste au calcul de l'exponentiation modulaire par le présent algorithme et permet d'éviter l'inconvénient cité juste ci-dessus.

On utilise deux registres R_1 et R_2 et un indicateur I qui est égal à zéro, " 0 ", signifiant que le résultat se trouve dans le registre R_1 ou qui est égal à un, " 1 ", signifiant que le résultat se trouve dans le registre R_2 ; ceci permet d'indiquer dans quel registre se trouve le bon résultat.

L'algorithme de l'invention qui reprend l'algorithme A consiste à s'exécuter par les étapes d'initialisation suivantes a et b et les étapes de calcul c, d , e et f qui sont effectuées k fois, k étant la taille de Y, étapes décrites ci-dessous:

a) On initialise $R_1=1$;

b) On initialise $I=0$;

Pour chaque bit Y(i) de la représentation binaire de Y, on effectue les quatre étapes c, d, e et f suivantes de " 0 " à " k-1 " ; on parcours la représentation binaire de Y du bit de poids fort Y(k-1) vers le bit de poids faible Y(0);

c) Si I=0, on effectue l'opération $R_2= (R_1)^2$; 2
Si I=1, on effectue l'opération $R_1= (R_2)^2$;

d) On complémente I, c'est à dire il change de valeur mais uniquement de " 0 " vers " 1 " ou de " 1 " vers " 0 " ;

e) On refait l'opération de test sur I :
si I=0, on effectue l'opération $R_2= R_1{*}X$;
si I=1, on effectue l'opération $R_1= R_2{*}X$;

f) Si Y(i) est égal à 1, alors on complémente I ;
si Y(i) est égal à 0, alors on garde I inchangé.

Ainsi, quelque soit la valeur de Y, on exécute toujours une opération de SQU et une opération de MUL. On aura donc à l'étape d l'une des deux signatures suivantes :

$S(R_2=SQU(R_1))$ ou $S(R_1=SQU(R_2))$.

On aura aussi à l'étape f une des deux signatures suivantes : $S(R_1=MUL(R_2,X))$ ou $S(R_2=MUL(R_1,X))$.

Les signatures de l'étape c sont équivalentes car elles utilisent les mêmes opérandes et effectuent la même opération (SQU).

Les signatures de l'étape e sont équivalentes car elles utilisent les mêmes opérandes et effectuent la même opération (MUL).

Par conséquent, il n'est plus possible de remonter à la valeur de Y qui sera une succession d'opérations (SQU) et (MUL).

L'application de la présente invention permet de calculer une exponentiation modulaire de manière sécurisée dans un composant électronique mettant en œuvre un algorithme à clé publique nécessitant un algorithme d'exponentiation modulaire.

L'algorithme de la présente invention qui reprend l'algorithme classique B consiste à s'exécuter par les étapes d'initialisation suivantes a et b et les étapes de calcul suivantes c, d et e qui sont effectuées k fois, k étant la taille de Y:

a) on initialise $R_1=1$ et $Z=X$;

b) on initialise $I=0$;

Pour chaque bit de Y(i) de la représentation binaire de Y, on effectue les trois étapes c, d

et e, i variant de " 0 " à " k-1 " ; on parcours la représentation binaire de Y du bit de poids faible Y(0) vers le bit de poids fort Y(k-1) ;

c) on effectue l'opération : $Z:=Z^2$;

d) si I=0, on effectue l'opération : $R_2:=R_1*Z$;
si I=1, on effectue l'opération : $R_1:=R_2*Z$;

e) si Y(i)=0, alors on garde I inchangé et si Y(i)=1, alors on complémente I.

Ainsi, quelque soit la valeur de Y, on exécute toujours une opération de SQU et une opération MUL. On aura donc à l'étape c la signature suivante : S(SQU).
On aura à l'étape d une des deux signatures suivantes : $S(R_1=MUL(R_2,Z))$ ou $S(R_2=MUL(R_1,Z))$.
Les signatures de l'étape d sont équivalentes car elles utilisent les mêmes opérandes et effectuent la même opération (MUL).
Par conséquent, il n'est plus possible de remonter à la valeur Y qui sera une succession d'opérations (SQU) et (MUL). L'application de la présente invention permet de calculer une exponentiation modulaire de manière sécurisée dans un composant électronique mettant en œuvre un algorithme à clé publique nécessitant un algorithme d'exponentiation modulaire.

Comme exemple de l'invention, on utilise le DSA qui est une variante de l'algorithme de signatures de Schnorr et ElGamal.
Pour signer les étapes m, les étapes suivantes sont réalisées :

1) Génération d'un nombre aléatoire K ;

2) Calcul de $r=(g^k modp) modq$
avec g, p et q des nombres entiers publics connus par le monde extérieur de la carte à puce ;

3) Calcul de $s=(K^{-1}(H(m)+x*r))modq$
avec H() fonction de hachage et x une clé privée .

Le couple (r,s) correspond à la signature du message m.

On remarque que K est secret.

L'étape 2 consiste en partie à une exponentiation modulaire :

$$r'=g^K modp \text{ et } r=r'modq.$$

Si l'exponentiation modulaire est effectuée avec l'algorithme classique A ou B comme décrit précédemment, alors une attaque SPA permet de remonter à la valeur k. Connaissant k et comme s, m et r sont connus, l'attaquant peut calculer la clé secrète x. Ainsi, il a trouvé la clé de la signature et le système est cassé. Il est donc préférable d'utiliser la présente invention ou sa variante de réalisation pour effectuer l'exponentiation modulaire de l'étape 2 du présent exemple.

Ainsi, dans la présente invention, la méthode de calcul de l'algorithme ne permettant pas de retrouver k par étude de la consommation de courant , l'attaquant ne peut pas remonter à la valeur de la clé privée x.

REVENDICATIONS

1- Algorithme d'exponentiation modulaire défini par la formule mathématique suivante :

$R = X^Y \bmod N$,

Y étant un exposant présentant une taille de k bits ,
N étant un modulus présentant une taille de k' bits,
X étant une variable connue présentant une taille de k'' bits ;
R étant le résultat de l'opération d'exponentiation modulaire et présentant une taille de k' bits
et
comprenant des registres R1 et R2 et un indicateur I, algorithme caractérisé en ce qu'il présente les étapes d'exécution suivantes comprenant les étapes a et b, dites étapes d'initialisation, et les étapes c, d et e, dites étapes de calcul :
a) on initialise $R_1=1$ et $Z=X$;
b) on initialise $I=0$;
pour chaque bit de Y(i) de la représentation binaire de Y, on effectue les trois étapes c, d et e, i variant de " 0 " à " k-1 " ; on parcours donc la représentation binaire de Y du bit de poids faible Y(0) vers le bit de poids fort Y(k-1) ;

c) on effectue l'opération : $Z:=Z^2$;
d) si I=0, on effectue l'opération : $R_2:=R_1*Z$;
si I=1, on effectue l'opération : $R_1:=R_2*Z$;
e) si Y(i)=0, alors on garde I inchangé et si Y(i)=1, alors on complémente I.

2- Algorithme d'exponentiation modulaire défini par la formule mathématique suivante :

$R= X^Y \bmod N,$

Y étant un exposant présentant une taille de k bits ;
N étant un modulus présentant une taille de k' bits.
X étant une variable connue présentant une taille de k'' bits ;
R étant le résultat de l'opération d'exponentiation modulaire présentant une taille de k' bits
Et
comprenant des registres R1 et R2 et un indicateur I, algorithme caractérisé en ce qu'il présente les étapes d'exécution suivantes comprenant les étapes a et b, dites étapes d'initialisation, et les étapes c, d, e et f, dites étapes de calcul :
a) On initialise $R_1=1$;
b) On initialise I=0 ;
Pour chaque bit Y(i) de la représentation binaire de Y, on effectue les quatre étapes c,

d, e et f suivantes, i variant de " 0 " à " k-1 " , on parcours la représentation binaire de Y du bit de poids fort Y(k-1) vers le bit de poids faible Y(0);

c) Si I=0, on effectue l'opération $R_2 = (R_1)^2$;
Si I=1, on effectue l'opération $R_1 = (R_2)^2$;

d) Dans les deux cas de l'étape d, on complémente I, c'est à dire il change de valeur mais uniquement de " 0 " vers " 1 " ou de " 1 " vers " 0 " ;

e) On refait l'opération de test sur I :
si I=0, on effectue l'opération $R_2 = R_1 * X$;
si I=1, on effectue l'opération $R_1 = R_2 * X$;

f) Si Y(i) est égal à 1, alors on complémente I ;
si Y(i) est égal à 0, alors on garde I inchangé.

3- Composant électronique caractérisée en ce qu'il met en œuvre l'une quelconque des revendications 1 à 2.

4-Composant électronique selon la revendication 2 caractérisé en ce qu'il est un objet portable électronique du type carte à puce.

5- Terminal électronique caractérisé en ce s'il met en œuvre l'une quelconque des revendications 1 à 2.